Übungen zum Kapitel 8

Datenaufbereitung - Verknüpfen und Umformen

Erstellt und überarbeitet: armin.baenziger@zhaw.ch, 28. Februar 2020

In [1]:
```
%autosave 0
```

```
Autosave disabled
```

(A.1) Laden Sie NumPy und Pandas mit der üblichen Abkürzung.

In [2]:
```
import numpy as np
import pandas as pd
```

(A.2) Laden Sie die Daten der Datei `drinksbycountry.csv` in das DataFrame `drinks` und lesen Sie die ersten 5 Zeilen aus. Die Datei befindet sich im Ordner "weitere_Daten".

In [3]:
```
drinks = pd.read_csv('../weitere_Daten/drinksbycountry.csv')
drinks.head()
```

Out[3]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	continent
0	Afghanistan	0	0	0	0.0	Asia
1	Albania	89	132	54	4.9	Europe
2	Algeria	25	0	14	0.7	Africa
3	Andorra	245	138	312	12.4	Europe
4	Angola	217	57	45	5.9	Africa

(A.3) Erstellen Sie für das DataFrame `drinks` einen hierarchischen Index (mit `set_index`), wobei die erste Hierarchieebene der Kontinent (`continent`) und die zweite Ebene das Land (`country`) sein soll.

In [4]:
```
drinks = drinks.set_index(['continent', 'country'])
drinks.head()
```

Out[4]:

		beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
continent	**country**				
Asia	**Afghanistan**	0	0	0	0.0
Europe	**Albania**	89	132	54	4.9
Africa	**Algeria**	25	0	14	0.7
Europe	**Andorra**	245	138	312	12.4
Africa	**Angola**	217	57	45	5.9

(A.4) Sortieren Sie den (hierarchischen) Index des DataFrames `drinks` . Verwenden Sie hierzu die Methode `sort_index(inplace=True)` .

In [5]:
```
drinks.sort_index(inplace=True)
drinks.head()
```

Out[5]:

continent	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
Africa	**Algeria**	25	0	14	0.7
	Angola	217	57	45	5.9
	Benin	34	4	13	1.1
	Botswana	173	35	35	5.4
	Burkina Faso	25	7	7	4.3

(A.5) Lesen Sie die ersten 5 Zeilen (`head`) des Kontinents Europa ("Europe") aus.

In [6]:
```
drinks.loc['Europe'].head()
```

Out[6]:

country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
Albania	89	132	54	4.9
Andorra	245	138	312	12.4
Armenia	21	179	11	3.8
Austria	279	75	191	9.7
Azerbaijan	21	46	5	1.3

Bevor die nächsten Aufgaben gestellt werden, generieren wir zwei DataFrames:

In [7]:
```
# Betrachten Sie einfach das Ergebnis:
np.random.seed(11)
date = pd.date_range(start='2018', freq='Q', periods=8)
df2018 = pd.DataFrame(np.random.randn(4,2), index=date[:4],
                      columns=['A', 'B'])
df2018
```

Out[7]:

	A	B
2018-03-31	1.749455	-0.286073
2018-06-30	-0.484565	-2.653319
2018-09-30	-0.008285	-0.319631
2018-12-31	-0.536629	0.315403

In [8]:
```
df2019 = pd.DataFrame(np.random.randn(4,2), index=date[-4:],
                      columns=['A', 'B'])
df2019
```

Out[8]:

	A	B
2019-03-31	0.421051	-1.065603
2019-06-30	-0.886240	-0.475733
2019-09-30	0.689682	0.561192
2019-12-31	-1.305549	-1.119475

(B.1) Fügen Sie die Daten für das Jahr 2018 und 2019 in ein DataFrame mit Name `df` zusammen. Tipp: Im Pandas-Cheat-Sheet nachsehen.

In [9]:
```
df = pd.concat([df2018, df2019])
df
```

Out[9]:

	A	B
2018-03-31	1.749455	-0.286073
2018-06-30	-0.484565	-2.653319
2018-09-30	-0.008285	-0.319631
2018-12-31	-0.536629	0.315403
2019-03-31	0.421051	-1.065603
2019-06-30	-0.886240	-0.475733
2019-09-30	0.689682	0.561192
2019-12-31	-1.305549	-1.119475

(B.2) *Repetition*: Ersetzen Sie im DataFrame `df2` , welches eine *Kopie* von `df` ist, alle *negativen* Werte mit `0` .

In [10]:
```
df2 = df.copy()
df2[df2<0] = 0
df2
```

Out[10]:

	A	B
2018-03-31	1.749455	0.000000
2018-06-30	0.000000	0.000000
2018-09-30	0.000000	0.000000
2018-12-31	0.000000	0.315403
2019-03-31	0.421051	0.000000
2019-06-30	0.000000	0.000000
2019-09-30	0.689682	0.561192
2019-12-31	0.000000	0.000000

(B.3) *Repetition*: Ersetzen Sie im DataFrame `df3` , welches eine *Kopie* von `df` ist, alle *negativen* Werte *in Spalte* `B` mit `0` .

In [11]:
```
df3 = df.copy()
df3.B[df3.B < 0] = 0
df3
```

Out[11]:

	A	B
2018-03-31	1.749455	0.000000
2018-06-30	-0.484565	0.000000
2018-09-30	-0.008285	0.000000
2018-12-31	-0.536629	0.315403
2019-03-31	0.421051	0.000000
2019-06-30	-0.886240	0.000000
2019-09-30	0.689682	0.561192
2019-12-31	-1.305549	0.000000

Für das Verknüpfen von Daten laden wir zuerst das DataFrame `dflohn` (Verzeichnis "weitere_Daten"), welches wir bereits zuvor gebraucht haben:

In [12]:
```
# Beispieldaten laden:
dflohn = pd.read_pickle('../weitere_Daten/dflohn.pkl').reset_index()
dflohn.tail()
```

Out[12]:

	Person	Lohn	Geschlecht	Alter	Zivilstand
95	96	7959.0	w	55	v
96	97	2673.0	m	18	g
97	98	4430.0	m	19	v
98	99	6366.0	w	21	g
99	100	6488.0	m	24	g

(C.1) *Repetition:* Erstellen Sie eine absolute Häufigkeitstabelle der Spalte `Zivilstand`.

In [13]:
```
dflohn.Zivilstand.value_counts()
```

Out[13]:
```
vw    28
v     27
g     25
l     20
Name: Zivilstand, dtype: int64
```

Angenommen wir haben zusätzliche Informationen zu einem Teil der Personen, welche in `dfsektor` (wird in nächster Zelle erstellt) gespeichert sind:

In [14]:
```
# Beispieldaten generieren:
m=50  # Anzahl Beobachtungen
np.random.seed(11)
s1 = (pd.Series(dflohn.index, name='Person')     # Erstelle Series aus dflohn.index (Personen-Index)
          .sample(m)                             # Wähle aus den 100 Personen m ohne Zurücklegen.
          .reset_index(drop=True)                # Neuer durchlaufender Index (alter Index nicht in Spalte)
     )
s2 = (pd.Series(['primär', 'sekundär', 'teritär'], name='Sektor') # Sektor, in welchem Person tätig ist
      .sample(50, weights=[0.05, 0.30, 0.65], replace=True)       # Ziehen mit Zurücklegen mit untersch. W'keiten
      .reset_index(drop=True)
     )
dfsektor = pd.concat([s1, s2], axis=1)
dfsektor.tail()
```

Out[14]:

	Person	Sektor
45	77	teritär
46	72	teritär
47	8	sekundär
48	3	teritär
49	59	teritär

(C.2) Verknüpfen Sie die zwei DataFrames `dflohn` und `dfsektor` über die Spalte `Person` derart, dass zu denjenigen Personen in `dflohn`, zu denen es einen Match in `dfsektor` gibt, die entsprechenden Werte dem Frame `dflohn` hinzugefügt werden. Tipp: `pd.merge` und Cheat-Sheet verwenden.

In [15]:
```
pd.merge(dflohn, dfsektor,
         left_on='Person', right_on='Person',
         how='left').tail()
```

Out[15]:

	Person	Lohn	Geschlecht	Alter	Zivilstand	Sektor
95	96	7959.0	w	55	v	teritär
96	97	2673.0	m	18	g	teritär
97	98	4430.0	m	19	v	sekundär
98	99	6366.0	w	21	g	sekundär
99	100	6488.0	m	24	g	NaN

In diesem Fall geht es auch einfacher, da Pandas eine einzige gemeinsame Spalte `Person` in beiden DataFrames erkennt.

In [16]:
```
pd.merge(dflohn, dfsektor, how='left').tail()
```

Out[16]:

	Person	Lohn	Geschlecht	Alter	Zivilstand	Sektor
95	96	7959.0	w	55	v	teritär
96	97	2673.0	m	18	g	teritär
97	98	4430.0	m	19	v	sekundär
98	99	6366.0	w	21	g	sekundär
99	100	6488.0	m	24	g	NaN

(C.3) Ersetzten Sie im DataFrame `dflohn` den Index mit der Spalte `Person` (permanent). Tipp: Methode `set_index` mit dem Argument `inplace=True` verwenden.

In [17]:
```
dflohn.set_index('Person', inplace=True)
dflohn.head()
```

Out[17]:

Person	Lohn	Geschlecht	Alter	Zivilstand
1	4107.0	m	40	g
2	5454.0	m	47	vw
3	3719.0	m	41	g
4	6194.0	m	18	v
5	NaN	m	27	v

(C.4) Ersetzten Sie im DataFrame `dfsektor` den Index mit der Spalte `Person` (permanent).

In [18]:
```
dfsektor.set_index('Person', inplace=True)
dfsektor.head()
```

Out[18]:

Person	Sektor
46	sekundär
49	teritär
22	sekundär
58	teritär
41	sekundär

(C.5) Verknüpfen Sie die zwei DataFrames `dflohn` und `dfsektor` über den Index `Person` derart, dass zu denjenigen Personen in `dflohn` , zu denen es einen Match in `dfsektor` gibt, die entsprechenden Werte dem Frame `dflohn` hinzugefügt werden. Da der Match-Key in beiden Fällen der Index ist, eignet sich hierfür die Methode `join` besonders.

In [19]:
```
# Lösung mit join:
dflohn.join(dfsektor).tail()
```

Out[19]:

Person	Lohn	Geschlecht	Alter	Zivilstand	Sektor
96	7959.0	w	55	v	teritär
97	2673.0	m	18	g	teritär
98	4430.0	m	19	v	sekundär
99	6366.0	w	21	g	sekundär
100	6488.0	m	24	g	NaN

(D.1) Formen Sie das DataFrame `drinks` wie folgt um:

```
drinks.unstack()
```

Warum ist die Darstellung dieser Daten im "Wide-Format" nicht sinnvoll?

In [20]:
```
drinks.unstack()
# Nicht sinnvoll, da jedes Land nur einem Kontinent zugeordnet ist.
# Somit gibt es sehr viele NaN.
```

Out[20]:

	beer_servings									
country	**Afghanistan**	**Albania**	**Algeria**	**Andorra**	**Angola**	**Antigua & Barbuda**	**Argentina**	**Armenia**	**Australia**	**Austria**
continent										
Africa	NaN	NaN	25.0	NaN	217.0	NaN	NaN	NaN	NaN	NaN
Asia	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Europe	NaN	89.0	NaN	245.0	NaN	NaN	NaN	21.0	NaN	279.0
North America	NaN	NaN	NaN	NaN	NaN	102.0	NaN	NaN	NaN	NaN
Oceania	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	261.0	NaN
South America	NaN	NaN	NaN	NaN	NaN	NaN	193.0	NaN	NaN	NaN

6 rows × 772 columns

Ende der Übung